Weekly Report #3

Group Nr. 21: "Face Reconstruction"

February 15, 2022

1 Weekly Progress

Face Detection: Using the OpenCV library, we are able to locate the face in the image for an initial registration. We refer to Vikas Gupta's blog[2], where the performance and accuracy of several popular face detection methods are compared in detail. It is concluded that for high resolution images (Our Dataset has ColorMap with resolution of 1920 × 1080) and also in general cases, OpenCV DNN has a salient recognition performance and inference accuracy.

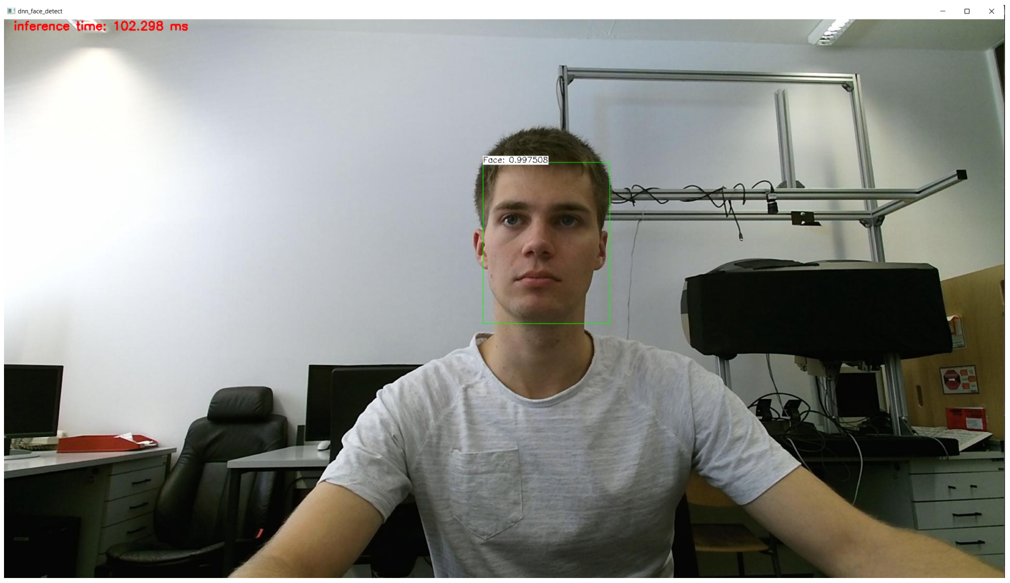


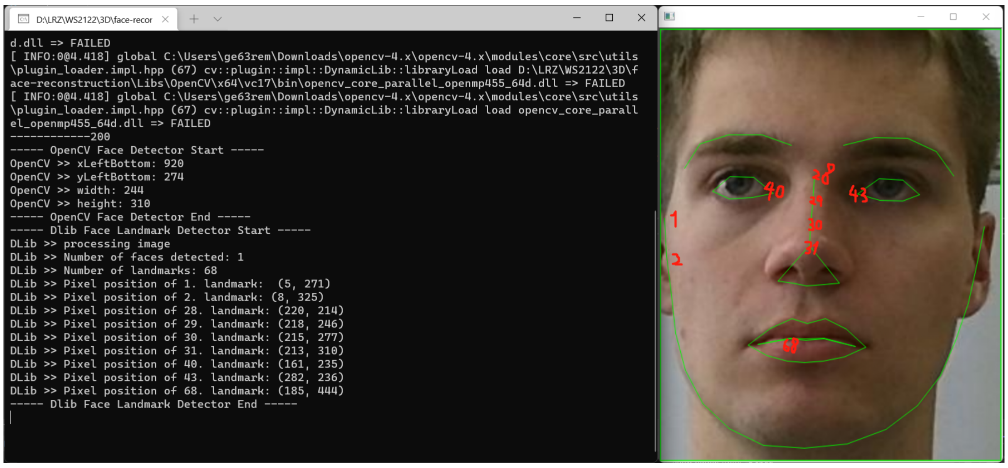
Figure 1: The performance and accuracy of OpenCV DNN Face Detection

As shown in Figure 1, for a high-resolution image of 1920 × 1080, the method takes only about 102ms to give the area where the face is located (the chartreuse rectangle box), and the accuracy of the inference is about 99.75%.

Landmark Detection: In this step, the DLib library is employed to get the pixel position of some crucial facial landmarks. We obtained the face area through the OpenCV DNN Face Detector, and more specifically, we obtained a `cv::Mat` object of the image area, which can be accepted by DLib's Face Landmark Detector.

The reason we do not feed the original high-resolution image directly into Face Landmark Detector is that DLib landmark detector is not suitable for finding landmarks in high-resolution images (due to sliding window detection scheme), which can be extremely slow (usually greater than 30 seconds), so we first get the face subarea via OpenCV and then pass that into the DLib landmark detector, in this way, we just need roughly 3 seconds to accomplish both face area detection and landmark detection.

According to DLib's example code[1], they use a trained model[4] from Sagonas Christos et al. to perform the detection, there are 68 specified feature landmarks defined in the model, so the final result of the detection will be the pixel coordinates of those face landmarks. The location of these 68 feature points on the face can be found here [1].



```
d.dll => FAILED
[ INFO:0@4.418] global C:\Users\ge63rem\Downloads\opencv-4.x\opencv-4.x\modules\core\src\utils
\plugin_loader.impl.hpp (67) cv::plugin::impl::DynamicLib::libraryLoad load D:\LRZ\WS2122\3D\f
ace-reconstruction\Libs\OpenCV\x64\vc17\bin\opencv_core_parallel_openmp455_64d.dll => FAILED
[ INFO:0@4.418] global C:\Users\ge63rem\Downloads\opencv-4.x\opencv-4.x\modules\core\src\utils
\plugin_loader.impl.hpp (67) cv::plugin::impl::DynamicLib::libraryLoad load opencv_core_parall
el_openmp455_64d.dll => FAILED
------------200
----- OpenCV Face Detector Start -----
OpenCV >> xLeftBottom: 920
OpenCV >> yLeftBottom: 274
OpenCV >> width: 244
OpenCV >> height: 310
----- OpenCV Face Detector End -----
----- Dlib Face Landmark Detector Start -----
DLib >> processing image
DLib >> Number of faces detected: 1
DLib >> Number of landmarks: 68
DLib >> Pixel position of 1. landmark:  (5, 271)
DLib >> Pixel position of 2. landmark: (8, 325)
DLib >> Pixel position of 28. landmark: (220, 214)
DLib >> Pixel position of 29. landmark: (218, 246)
DLib >> Pixel position of 30. landmark: (215, 277)
DLib >> Pixel position of 31. landmark: (213, 310)
DLib >> Pixel position of 40. landmark: (161, 235)
DLib >> Pixel position of 43. landmark: (282, 236)
DLib >> Pixel position of 68. landmark: (185, 444)
----- Dlib Face Landmark Detector End -----
```

Figure 2: The output of DLib Face Landmark Detector

As illustrated in Figure 2, the program successfully gives the starting position of the face area as well as its width and height. Moreover, we can also observe that the pixel coordinates of the output landmarks are also correct.

Face Model: We have started to look into the Basel Face Model which we currently plan to use as a parametric model for face tracking.[2]

[1]Facial point annotations - `https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/`
[2]`https://faces.dmi.unibas.ch/bfm/`

Work Distribution:

- Tong Yan Chan (03722291): Research on parametric face model & alignment
- Dushyant Anirudhdhabhai Dave (03728740): Research on face model & alignment
- Daniel Schubert (03666228): Research on Kinect RGB & D alignment
- Chang Luo (03759570): Integrate OpenCV DNN with DLib to speed up face landmark detection

2 Problems

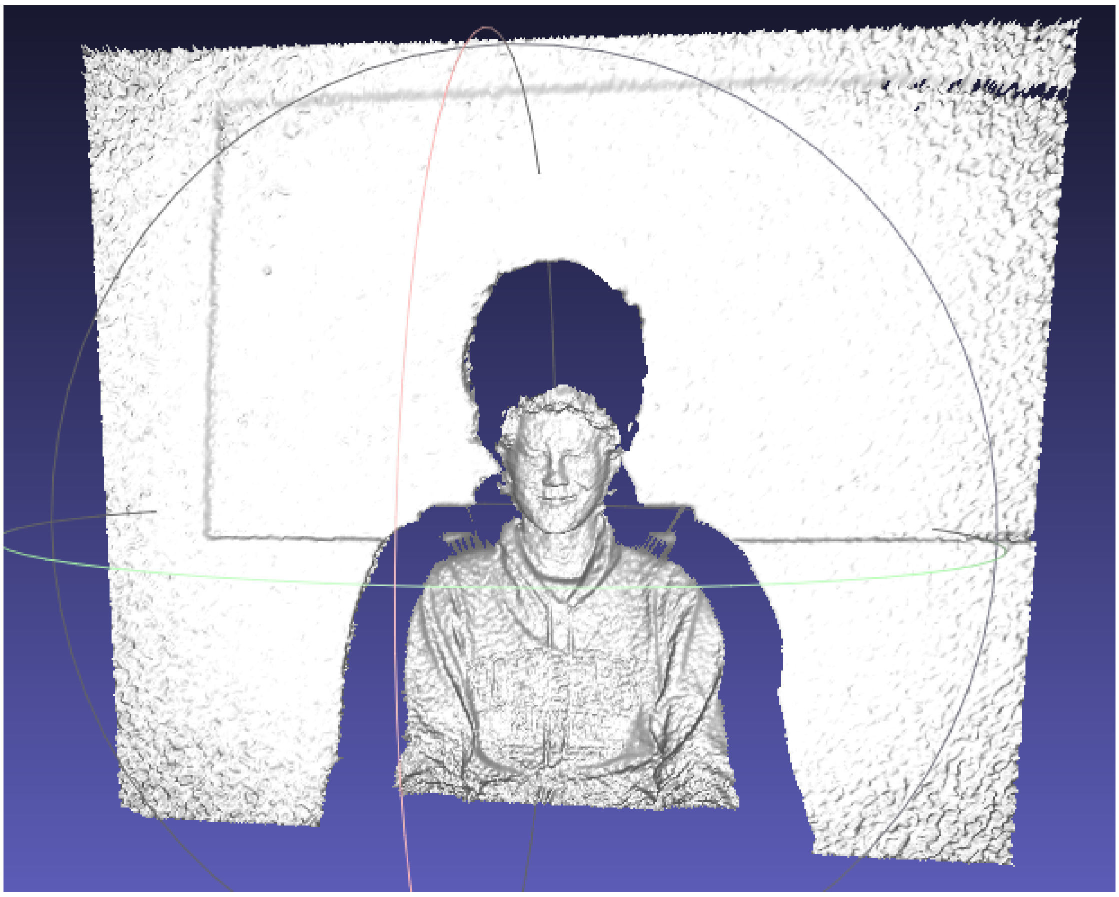


Figure 3: Point cloud computed by the depth images.

Dataset: As shown in Fig.3, we have already set up a pipeline to compute the point cloud from the images obtained by Kinect in the FaceGrabber dataset [3] last week. We plan to cut the face from the point cloud this week. Unfortunately, there is an alignment issue on between the depth images and the color images.

The largest alignment problem stems from the fact that the depth camera has a large offset in space on the Kinect V2.

This gives it a different perspective, which in turn breaks any alignment based on simple 2D transform for objects close to the camera. Therefore, we cannot cut the face out directly on the depth map according to the coordinates on the colour image where we detect a face. We are still in progress to resolve this issue.

We are investigating a workaround on this issue. Since most of our algorithm will be only depth data, we hope that a non-perfect alignment will not affect the reconstruction dramatically. However, we have not yet fully investigated the impact it might have on the ICP optimisation.

3 Plan

The next steps we plan for our project are:

- Align the depth map with color map
- Initial registration of Face model on the 3D point cloud
- Apply Rigid ICP as the initial state

References

[1] dlib. dlib C++ Library - face_landmark_detection_ex.cpp.

[2] Vikas Gupta. Face Detection OpenCV, Dlib and Deep Learning (C++ / Python).

[3] D. Merget, T. Eckl, M. Schwrer, P. Tiefenbacher, and G. Rigoll. Capturing facial videos with kinect 2.0: A multithreaded open source tool and database. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

[4] Christos Sagonas, Epameinondas Antonakos, Georgios Tzimiropoulos, Stefanos Zafeiriou, and Maja Pantic. 300 Faces In-The-Wild Challenge: database and results. *Image and Vision Computing*, 47:3–18, March 2016.